Task:1 Implement the Linear regression and gradient descent from scratch using numpy..

```
import numpy as np
```

- dl/dm = -2x(y-y^)/n
- dl/db = -2(y-y^)/n

```
class MyLinearRegression():
    def __init__(self, learning_rate=0.0001, epochs=100, batch_size=32):
        self.learning_rate = learning_rate
        self.epochs = epochs
        self.batch_size = batch_size
        self.coef_ = None
        self.intercept_ = None

    def fit(self, x, y):
        n_samples, n_features = x.shape
        self.coef_ = np.zeros(n_features)  # Initialize coefficients to zeros
        self.intercept_ = 0

        # Standardize the data
        x = (x - np.mean(x, axis=0)) / np.std(x, axis=0)
        y = (y - np.mean(y)) / np.std(y)

        for epoch in range(self.epochs):
            indices = np.arange(n_samples)
            np.random.shuffle(indices)  # Shuffle indices for better randomness

            for i in range(0, n_samples, self.batch_size):
                x_batch = x[indices[i:i + self.batch_size]]
                y_batch = y[indices[i:i + self.batch_size]]

                y_hat = np.dot(x_batch, self.coef_) + self.intercept_

                dl_dm = -2 * np.dot((y_batch - y_hat), x_batch) / len(y_batch)

                # Gradient clipping to prevent overflow
                dl_dm = np.clip(dl_dm, -1e2, 1e2)

                self.coef_ -= self.learning_rate * dl_dm

                dl_db = -2 * np.mean(y_batch - y_hat)
                dl_db = np.clip(dl_db, -1e2, 1e2)

                self.intercept_ -= self.learning_rate * dl_db

            # Optionally print the loss at each epoch for monitoring
            if epoch % 10 == 0:
                loss = np.mean((y - self.predict(x)) ** 2)
                print(f'Epoch {epoch}: Loss = {loss:.4f}')
```

```
    def predict(self, x):
        # Standardize the input data using the same parameters as in fit
        x = (x - np.mean(x, axis=0)) / np.std(x, axis=0)
        return np.dot(x, self.coef_) + self.intercept_
```

```
from tensorflow.keras.datasets import california_housing
```

```
(x_train,y_train),(x_test,y_test) = california_housing.load_data()
print(x_train.shape,y_train.shape,x_test.shape,y_test.shape)
```

```
(16512, 8) (16512,) (4128, 8) (4128,)
```

```
mylr = MyLinearRegression(learning_rate = 0.01, epochs = 50, batch_size = 32)
mylr.fit(x_train,y_train)
```

```
Epoch 0: Loss = 0.3841
Epoch 10: Loss = 0.3623
Epoch 20: Loss = 0.3859
Epoch 30: Loss = 0.3626
Epoch 40: Loss = 0.3617
```

```
mylr.predict(x_test)
```

```
array([ 0.15800416,  0.5896171 , -0.08612715, ..., -0.68536264,
        -0.11347314,  0.03640464])
```

```
print(y_test)
```

```
[397900. 227900. 172100. ...  98800. 234600. 100000.]
```

Task:2 Use the scikit-learn library for linear regression and explore that on california housing price prediction.

```
from sklearn.linear_model import LinearRegression
```

```
lr1 = LinearRegression()
lr1.fit(x_train,y_train)
```

```
LinearRegression()
```

```
lr1.predict(x_test)
```

```
array([222345.75, 277794.  , 198939.75, ..., 125391.75, 188751.25,
       205716.75], dtype=float32)
```

```
print(y_test)
```

```
[397900. 227900. 172100. ...  98800. 234600. 100000.]
```

```
from sklearn.metrics import mean_squared_error, r2_score

# For the scikit-learn model on California housing
y_pred_sklearn_cali = lr1.predict(x_test)
mse_sklearn_cali = mean_squared_error(y_test, y_pred_sklearn_cali)
r2_sklearn_cali = r2_score(y_test, y_pred_sklearn_cali)

print("Scikit-learn California Housing - Mean Squared Error:", mse_sklearn_cali)
print("Scikit-learn California Housing - R-squared:", r2_sklearn_cali)
```

```
Scikit-learn California Housing - Mean Squared Error: 4940737000.0
Scikit-learn California Housing - R-squared: 0.6283579882179436
```

Task: 3 Use the scikit-learn library for multiple linear regression and explore HR dataset.

```
import pandas as pd
```

```
df = pd.read_csv('/content/HR_comma_sep.csv')
df.head()
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_comp
0	0.38	0.53	2	157	
1	0.80	0.86	5	262	
2	0.11	0.88	7	272	
3	0.72	0.87	5	223	
4	0.37	0.52	2	159	

```
y = df.iloc[:,-1]
y.head()
```

	salary
0	low
1	medium
2	medium
3	low
4	low

dtype: object

```
x = df.iloc[:,:-1]
x.head()
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_comp
0	0.38	0.53	2	157	
1	0.80	0.86	5	262	
2	0.11	0.88	7	272	
3	0.72	0.87	5	223	
4	0.37	0.52	2	159	

```
x['Department'].unique()
```

```
array(['sales', 'accounting', 'hr', 'technical', 'support', 'management',
       'IT', 'product_mng', 'marketing', 'RandD'], dtype=object)
```

```
x['Department'].value_counts()
```

Department	count
sales	4140
technical	2720
support	2229
IT	1227
product_mng	902
marketing	858
RandD	787
accounting	767
hr	739
management	630

dtype: int64

```
x = pd.get_dummies(x,columns=['Department'],drop_first=True)
```

```
x.head()
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_comp
0	0.38	0.53	2	157	
1	0.80	0.86	5	262	
2	0.11	0.88	7	272	
3	0.72	0.87	5	223	
4	0.37	0.52	2	159	

```
# prompt: apply label encoding on y which is target column

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
```

```
y[0:5]
```

```
array([1, 2, 2, 1, 1])
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

```
print(x_train.shape,y_train.shape,x_test.shape,y_test.shape)
```

```
(11999, 17) (11999,) (3000, 17) (3000,)
```

```
lr2 = LinearRegression()
lr2.fit(x_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
lr2.predict(x_test)
```

```
array([1.35747287, 1.38366086, 1.34804691, ..., 1.35417743, 1.34747087,
       1.36987414])
```

```
print(y_test)
```

```
[2 1 1 ... 2 2 1]
```

```
# For the scikit-learn model on HR dataset
y_pred_sklearn_hr = lr2.predict(x_test)
mse_sklearn_hr = mean_squared_error(y_test, y_pred_sklearn_hr)
r2_sklearn_hr = r2_score(y_test, y_pred_sklearn_hr)

print("Scikit-learn HR Dataset - Mean Squared Error:", mse_sklearn_hr)
print("Scikit-learn HR Dataset - R-squared:", r2_sklearn_hr)
```

```
Scikit-learn HR Dataset - Mean Squared Error: 0.39043561363255214
Scikit-learn HR Dataset - R-squared: 0.006693655955176014
```

Start coding or generate with AI.